Content to be covered:

What is the Goal of Angular 2?

Pre-requisite for Angular 2 before you start

Step 1:- Installing VS Code, Typescript and Node

With modularity comes great responsibility

Step 2:- Setting up Angular environment

Step 3:- Configuring the task runner

Understanding Angular 2 Component and module architecture

Step4:- Following MVW Step by Step – Creating the folders

Step 5:- Creating the model

Step 6:- Creating the Component

Step 7:- Creating the Customer HTML UI – Directives and Interpolation

Step\_8:-\_Creating\_the\_Module

Step 9:-\_Creating\_the\_“Startup.ts”\_file

Step 10:- Invoking “Startup.ts” file using main angular page

Step 11:- Installing http-server and running the application

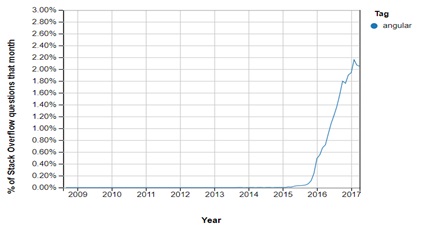
How to the run the source code?

## Acronym used in this notes

* NPM :- Node package manager.
* TS :- TypeScript.
* JS :- Javascript.
* VS :- Visual studio.
* VS Code :- Visual studio code.
* WP :- Webpack
* OS :- Operating system.

**Introduction**

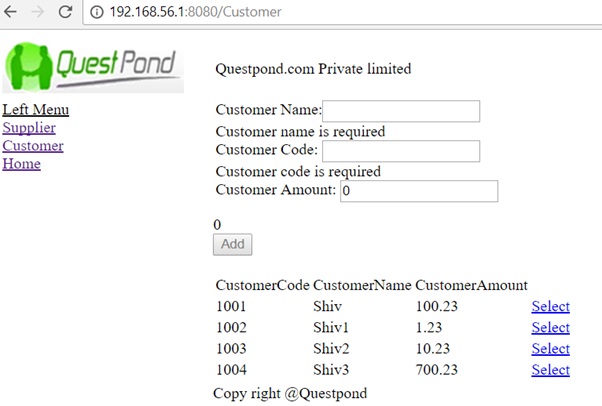
Why should we learn Angular? , the below stack overflow graph says it all. Its popular, its hot with lot of job openings. This article teaches you Angular step by step via 11 great Labs. So if you are here to learn Angular then you are at the right place and surfing the right article.

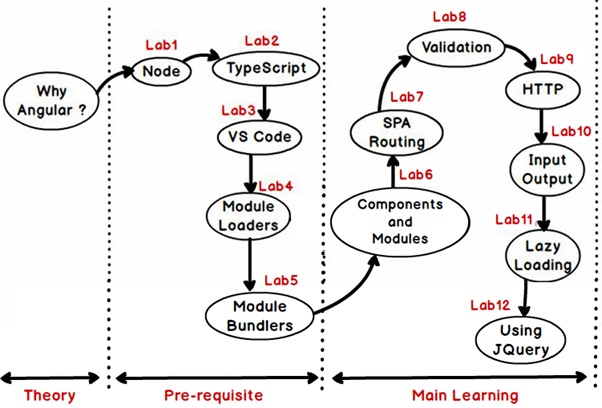


AngularJS vs Angular. There are two versions of Angular the old is named as AngularJS and the new one is named as just Angular. So when someone says AngularJS it means Angular 1.X and when someone says JUST Angular its Angular 2/4.

This project will have the following features:-

* Application should have capability of accepting three fields Customer name , Customer and Customer Amount values.
* Customer name and Customer codes are compulsory fields and it should be validated.
* Application will have a "Add" button which help us to post the current customer data to a Server. Once the data is added to the server it should displayed on the grid.
* Application will have a navigation structure wherein we will have logo and company name at the top , navigational link at the left and copy right details at the bottom of the screen.





So above is the road map of how this article will teach you angular. It has three phases: -

**Theory Phase:-** In this phase we will understand what is Angular and why do we need it.

**Pre-requisite phase :-** In this phase we will four important things Node, Typescript , VSCode , Module loaders ( SystemJS) and module bundlers ( Webpack).

**Main learning phase :-** This is where actual angular starts. In this we will be having 8 labs and while covering those labs we will creating the customer data entry screen project as discussed previously.

So do not wait any more start LAB by LAB and STEP by STEP.

★ **Should I start from Angular 1, 2 or 4.** Angular 1.X and 2.X are very much different. So even if you have done Angular 1.X you have to restart fresh from Angular 2.X. Angular 2.X and Angular 4.X are backward compatible so if you are learning Angular 2 you are learning Angular 4 and ahead. So people who are new to Angular just start from Angular 4 and **this article teaches Angular 4**.

#### Why do we need Angular ?

*"Angular is an open source JavaScript framework which simplifies* ***binding code*** *between JavaScript objects and HTML UI elements."*

Let us try to understand the above definition with simple sample code.

Below is a simple "Customer" function with "CustomerName" property. We have also created an object called as "Cust" which is of "Customer" class type.

function Customer()

{

this.CustomerName = "AngularInterview";

}

var Cust = new Customer();

Now let us say in the above customer object we want to bind to a HTML text box called as "TxtCustomerName". In other words when we change something in the HTML text box the customer object should get updated and when something is changed internally in the customer object the UI should get updated.

<input type=text id="TxtCustomerName" onChange="UitoObject()"/>

So in order to achieve this communication between UI to object developers end up writing functions as shown below. "UitoObject" function takes data from UI and sets it to the object while the other function "ObjecttoUI" takes data from the object and sets it to UI.

function UitoObject()

{

Cust.CustomerName = $("#TxtCustomerName").val();

}

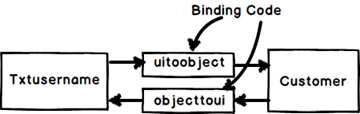
function ObjecttoUi()

{

$("#TxtCustomerName").val(Cust.CustomerName);

}

So if we analyze the above code visually it looks something as shown below. Your both functions are nothing but binding code logic which transfers data from UI to object and vice versa.

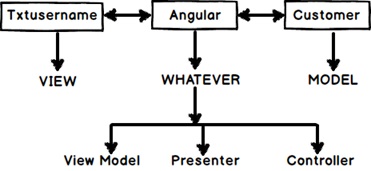


**Binding Code**

Now the same above code can be written in Angular as shown below. So now whatever you type in the textbox updates the "Customer" object and when the "Customer" object gets updated it also updates the UI.

<input type=text [(ngModel)]="Customer.CustomerName"/>

In short if you now analyze the above code visually you end up with something as shown in the below figure.You have the VIEW which is in HTML, your MODEL objects which are javascript functions and the binding code in Angular.



Now that binding code have different vocabularies.

* Some developers called it "ViewModel" because it connects the "Model" and the "View".
* Some call it "Presenter" because this logic is nothing but presentation logic.
* Some term it has "Controller" because it controls how the view and the model will communicate.

To avoid this vocabulary confusion Angular team has termed this code as "Whatever". It's that "Whatever" code which binds the UI and the Model. That's why you will hear lot of developers saying Angular implements "MVW" architecture.

This complete article is divided in to 10 Labs and each Lab is guided by detailed steps.

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/1.jpg | The best way to learn any new technology is by creating a project. So in this step by step series we will be creating a simple Customer data entry screen project.  So for Lab 1 we will create a basic screen in which we will create three fields “Name” , “Code” and “Amount”. When the end user types in these 3 fields the values will be displayed down below.  So in lab 1 we will execute 11 steps to achieve the same. |

Now this is a quiet big article so in case you want to go super-fast you can watch the below Angular 2 video. This video explains how to get your first Angular 2 video program running.

## What is the Goal of Angular 2?

*“Angular 2 is an open source JavaScript framework which simplifies binding code between JavaScript objects and HTML UI elements.”*

Let us try to understand the above definition with simple sample code.

Below is a simple “Customer” function with “CustomerName” property. We have also created an object called as “Cust” which is of “Customer” class type.

Hide   Copy Code

function Customer()

{

this.CustomerName = "AngularInterview";

}

var Cust = new Customer();

Now let us say in the above customer object we want to bind to a HTML text box called as “TxtCustomerName”. In other words when we change something in the HTML text box the customer object should get updated and when something is changed internally in the customer object the UI should get updated.

Hide   Copy Code

<input id="TxtCustomerName" onchange="UitoObject()" type="text" />

So in order to achieve this communication between UI to object developers end up writing functions as shown below. “UitoObject” function takes data from UI and sets it to the object while the other function “ObjecttoUI” takes data from the object and sets it to UI.

Hide   Copy Code

function UitoObject()

{

Cust.CustomerName = $("#TxtCustomerName").val();

}

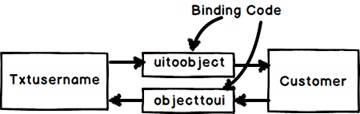
function ObjecttoUi()

{

$("#TxtCustomerName").val(Cust.CustomerName);

}

So if we analyze the above code visually it looks something as shown below. Your both functions are nothing but binding code logic which transfers data from UI to object and vice versa.



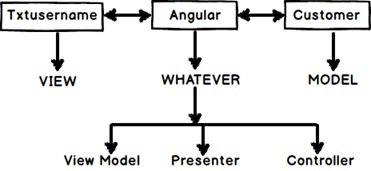
**Binding Code**

Now the same above code can be written in Angular as shown below. So now whatever you type in the textbox updates the “Customer” object and when the “Customer” object gets updated it also updates the UI.

Hide   Copy Code

<input type="text" />

In short if you now analyze the above code visually you end up with something as shown in the below figure.You have the VIEW which is in HTML, your MODEL objects which are javascript functions and the binding code in Angular.



**Javascript Functions**

Now that binding code have different vocabularies.

* Some developers called it “ViewModel” because it connects the “Model” and the “View” .
* Some call it “Presenter” because this logic is nothing but presentation logic.
* Some term it has “Controller” because it controls how the view and the model will communicate.

To avoid this vocabulary confusion Angular team has termed this code as “Whatever”. It’s that “Whatever” code which binds the UI and the Model. That’s why you will hear lot of developers saying Angular implements “MVW” architecture.

So concluding the whole goal of Angular is Binding, Binding and Binding.

## Pre-requisite for Angular 2 before you start

In my initial journey of learning Angular 2 I realized that Angular 2 by itself is easy to understand. But Angular 2 uses lot of JavaScript open sources and if you do not know these open sources Learning Angular 2 would become extremely difficult.

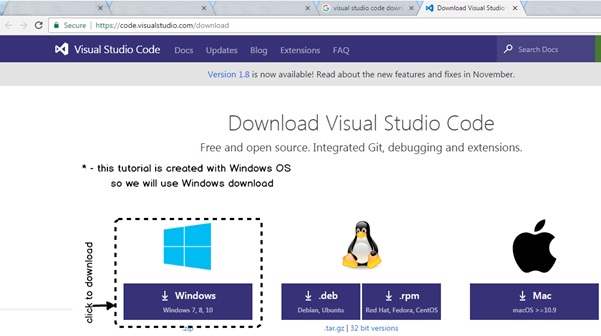
So we would recommend spending your some time going through the below videos and making notes around the same:-

|  |  |
| --- | --- |
| **Topic name** | **YouTube URL source** |
| Node JS | https://www.youtube.com/watch?v=-LF\_43\_Mqnw |
| Type Script | https://www.youtube.com/watch?v=xqYD8QXJX9U |
| System JS | https://www.youtube.com/watch?v=nQGhdoIMKaM |
| Common JS concept | https://www.youtube.com/watch?v=jN4IM5tp1SE |

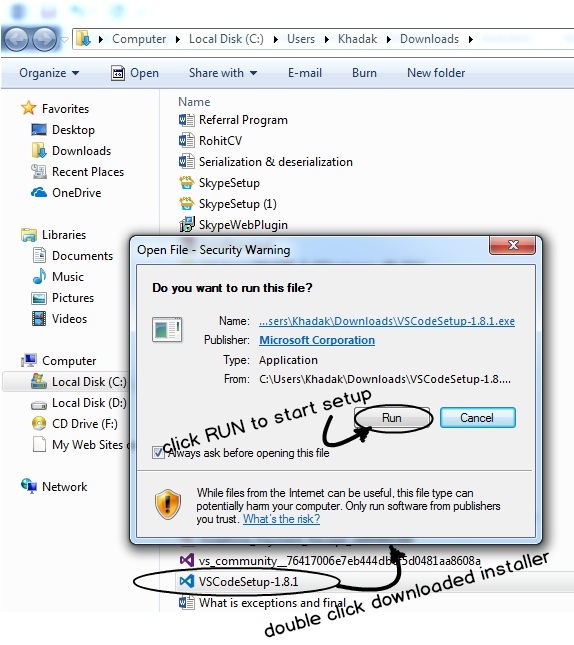
## Step 1:- Installing VS Code, Typescript and Node

Theoretically you can do Angular 2 with a simple notepad. But then that would be going back to back ages of adam and eve and reinventing the wheel. So in order to expedite the learning process we would be using some tools and software’s.

The first tool which we would need is Visual studio code editor or in short you can say VS code editor. So go to https://code.visualstudio.com/download and depending on your operating system install the appropriate one. For instance I am having windows OS so I will be installing the windows version.



Once you download the setup it’s a simple setup EXE run it and just hit next, next and finish.

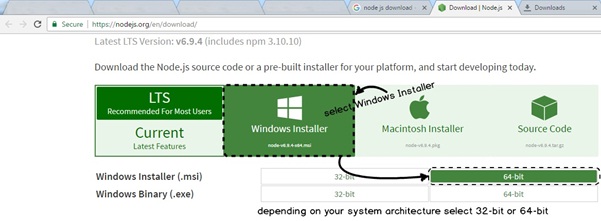


The second thing we need is “Nodejs”. We would encourage you to watch this video which explains the basic use of Node JS, Node JS Explained.

If I put in simple words NodeJs does the following two things:-

1. It has something called as “NPM” node package manager. If you watch around you will see lot of JS frameworks coming up every day. Each one of these frameworks have their own website, github repository with weird names and they are releasing new versions now and then. So to get these frameworks we need to go to their site download the JS files and so on. NPM is a central repository where all these things are registered. And as developer if you want to get a specific JS open source you can just type NPM commands like “npm install jquery”. This command will bring the recent version of Jquery in your computer folder.
2. The second thing which node does is it helps you to run JavaScript outside the browser. So tomorrow if you want to build a server application or windows application using JavaScript then this thing might help.

So to get node you can go to www.nodejs.org and depending on your OS select the appropriate download as shown in the below figure.



|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/7.jpg | Once you install node you should see NodeJs command prompt in your program files as shown in the figure.  We can then open the NodeJS command prompt and fire NPM commands and so on.  In case you are completely new to NodeJS please see this NodeJS Video which explains NodeJS in more details. |

The third thing for Angular 2 which we need is typescript. If I put in simple words:-

*“TypeScript is a sugar coated Object oriented programming language over JavaScript.”*

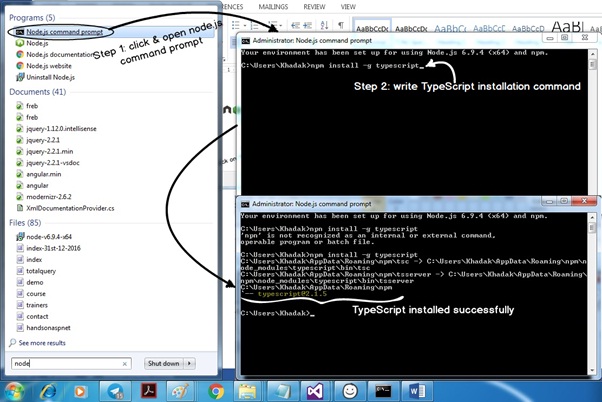
|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/8.jpg | So in typescript we can write code using keywords like “class” , “extends” , “interface” and so on.  Internally typescript will compile ( must be right word would be “transpile”) in to pure javascript code in terms of functions , closures and IIFE. |

Please do watch this 1 hour Training video on TypeScript which explains Typescript in more detail.

So as we said TypeScript is javascript open source we can get the same from the node. Remember in the previous section we discussed that node has NPM by which we can get latest versions of JS framework.

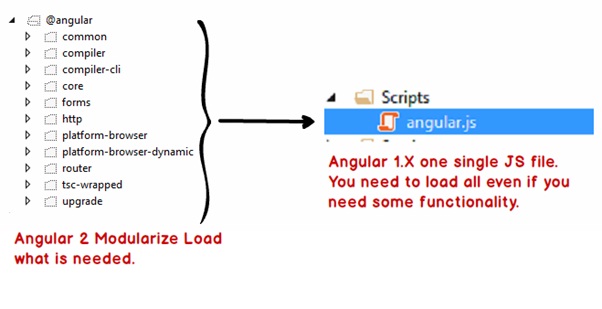
So click on node command prompt and in the command prompt type “npm install –g typescript”. This command will install typescript in your PC and make it available global throughout the computer.

**Note :-** The “-g” command makes typescript available throughout the computer from any folder command prompt.



So to start with Angular we need at least these three things one editor we have chosen VS Code , Node ( NPM) for getting JavaScript frameworks and Typescript so that we can code JavaScript faster and with Object Oriented coding approach.

## With modularity comes great responsibility



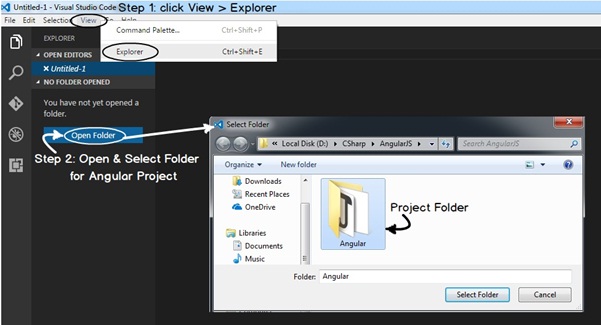
In Angular 1.X we just had one JS file which had the whole framework. So you drag and drop that single Angular 1.X JS file on HTML page and you are ready with the Angular 1.X environment. But the problem with having whole framework in one JS files was that you have to include all features even if you need it or not.

For instance if you are not using HTTP still that feature will be loaded. In case of Angular 2 we have separate discrete components. These are self-contained components which can be loaded in an isolated manner rather than loading the whole JS file.

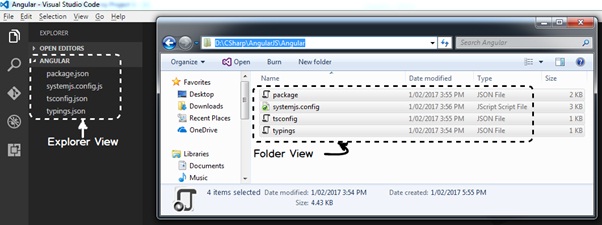
Now because we have lot of self-contained components in other words we have lot of single JS files creating a proper environment itself is a challenge. So let us first understand how to setup angular environment.

## Step 2:- Setting up Angular environment

So the first step is to create a folder and open the folder using VS Code as shown in the below image.



Inside this folder we need to paste the 4 JSON files. You can download these files from this URL http://tinyurl.com/jeehlqo.

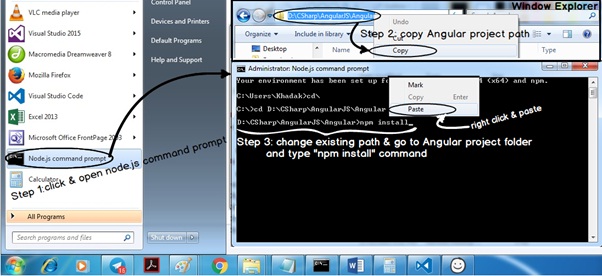


Please note these files are not created by me but I have got them from Angular 2 website or the respective open source website.

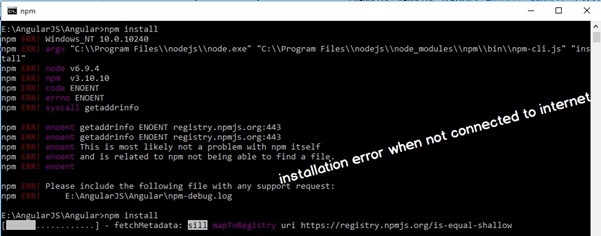
Below goes the explanation of those 4 JSON files:-

|  |  |
| --- | --- |
| **JSON File Name** | **What is the use of the file?** |
| package.json | This file has the references of all the Angular 2 modules. Node will use this file to download all the Angular modular files. If you are new to node please go through the pre-requisite video section. |
| tsconfig.json | This is a configuration file for typescript and will be used by typescript to define how transpiling process takes place in typescript. If you are not aware of typescript please go through the pre-requisite videos. |
| typings.json | Typescript is a new language some of the old JS frameworks cannot be consumed in typescript. For those frameworks we need to define the typings. |
| systemjs.config | SystemJS is a module loader. This configuration file defines the configuration for SystemJS module loader. We will be talking more about it in the further coming steps. |

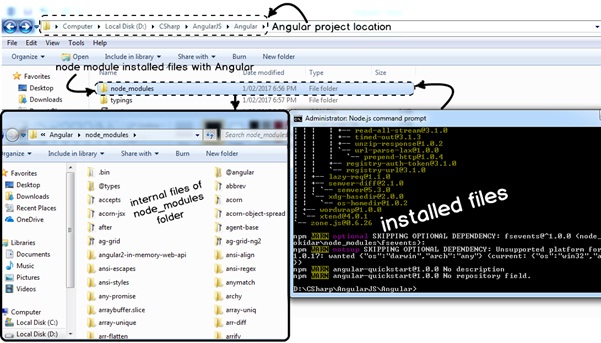
So now lets open “node” command prompt and type “npm install” in the command line. This command tries to get all the files which are mentioned in the package.json file.



You need to be connected to internet to get these files. If you are not connected to internet you would land up in to some kind of an error as shown in the below figure.



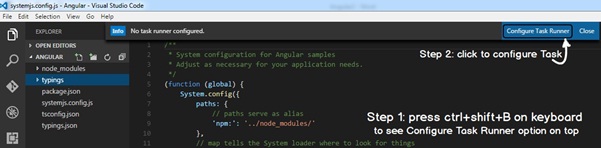
After the Angular installation is done successfully you will see “node\_modules” folder created. In this folder all the JS files of Angular 2 has been downloaded.



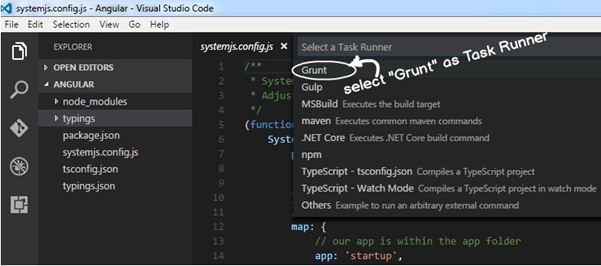
## Step 3:- Configuring the task runner

VS Code is just a code editor. It has no idea how to compile typescript code, how to run TSC and so on. So we need to create a GRUNT task which will run TSC command and transpile typescript code to JavaScript.

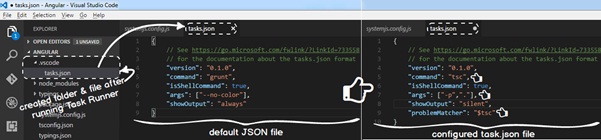
Press CONTROL + SHIFT + B and click on configure task runner as shown in the below figure.



VS Code then pops up lot of task runner options saying what kind of task is it, is it a GRUNT task, GULP task, MSBUILD, MAVEN etc. and so on.



Let’s select GRUNT Task and paste the below JSON code. Once you paste it you will see it has created a file called as “tasks.json” file inside “.vscode” folder.



Below is the paste of “tasks.json” file. You can see in the command attribute we have given “tsc” as the command line, the “isShellCommand” specifies that this has to be executed in command line and ‘args’ specifies the argument.

When typescript will execute it will run using the configuration specified in the tasks.json file.

Hide   Copy Code

{

// See https://go.microsoft.com/fwlink/?LinkId=733558

// for the documentation about the tasks.json format

"version": "0.1.0",

"command": "tsc",

"isShellCommand": true,

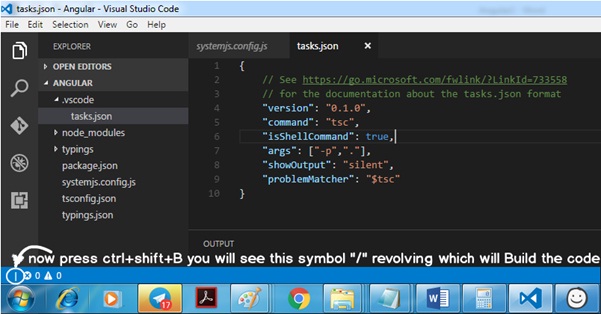
"args": ["-p","."],

"showOutput": "silent",

"problemMatcher": "$tsc"

}

So if you now press CONTROL +SHIFT + B it will build the TS files to JS files.



## Understanding Angular 2 Component and module architecture

As we said in the previous section that the whole goal of Angular 2 is binding the model and the view. In Angular 2 the binding code is officially termed as “Component”. So hence forth we will use the word “Component” for the binding code.

In enterprise projects you can have lot of components. With many components it can become very difficult to handle the project.

So you can group components logically in to modules.

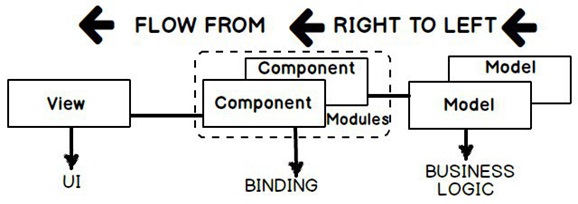
|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/20.jpg | So hence forth I will be using two terms:-  Components: - This will have the binding logic to bind the UI and the model.  Modules :- This will logically group components. |

## Step 4:- Following MVW Step by Step – Creating the folders

Before we start coding lets visualize the steps of coding. As we have said Angular 2 is a binding framework. It follows MVW architecture. It binds HTML UI with the JavaScript code (model).

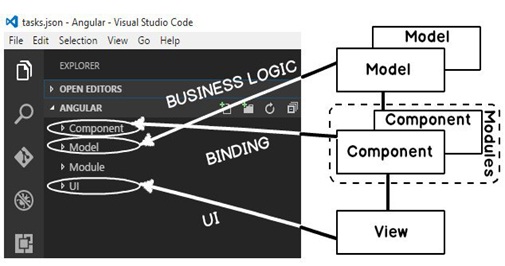
So if we visualize it will look something as shown in the image below. So let’s move from right to left. So let’s do the coding in the following sequence:-

1. Create the model.
2. Create the Component.
3. Create the module.
4. Create the HTML UI.

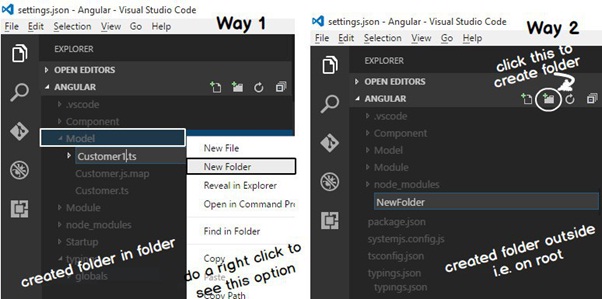


So let’s first create four folders in our project:-

* View folder: - This folder will contain the HTML UI.
* Model folder: - This folder will have the main business typescript classes.
* Component folder: - This folder will have the binding code which binds the HTML UI and Model.
* Module: - This folder will have code which will logically group the components.



In order to create a folder in VS code you can use the “New folder” icon or you can right click and also create a folder.



## Step 5:- Creating the model

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/24.jpg | A model is nothing but a class with properties and behavior. So let us first create the customer model with three properties “CustomerName”, “CustomerCode” and “CustomerAmount”.  So right click on the “Model” folder and add a new file “Customer.ts”. Keep the extension of this file as “.ts” as it’s a typescript file.  While compiling the typescript command identifies only files with the extension “.ts”. |

In the “Customer.ts” let’s create a “Customer” class with three properties. In this article we will not be going through the basics of typescript , please do go through this 1 hour training video of typescript which explains typescript in more detail.

Hide   Copy Code

export class Customer {

CustomerName: string = "";

CustomerCode: string = "";

CustomerAmount: number = 0;

}

## Step 6:- Creating the Component

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/25.jpg | The next thing we need to code is the binding code. Binding code in Angular 2 is represented by something termed as “COMPONENTS”. Angular 2 components has the logic which helps to bind the UI with the model.  So right click on the component folder and add “CustomerComponent.ts” file as shown in the figure at the left. |

In the component we need to import two things the Angular 2 core and our Customer model. Please note “import” is a typescript syntax and not JavaScript. So in case you are not following the code , please see this Learn Typescript in 1 hour video before moving ahead.

Hide   Copy Code

import {Customer} from '../Model/Customer'

import {Component} from "@angular/core"

The first line imports the “Customer” class in to the “CustomerComponent.ts”. This import is only possible because we have written “export” in “Customer.ts” file. The import and export generate code which follows CommonJs , AMD or UMD specifications. In case you are new to these specifications please see this CommonJs video which explains the protocol in more detail.

Hide   Copy Code

import {Customer} from '../Model/Customer'

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/26.jpg | Let’s try to understand how “Customer” component is located. If you see it’s using a relative path. In the import it says “../Model/Customer”.  The “../” says go one folder up. So we are currently in the “Component” folder , so with “../” it travels to the root and from that point it makes entry in to the “Model” folder. |

The next import command imports angular core components. In this we have not given any relative path using “../” etc. So how does typescript locate the angular core components ?.

Hide   Copy Code

import {Component} from "@angular/core"

If you remember we had used node to load Angular 2 and node loads the JS files in the “node\_modules” folder. So how does typescript compiler automatically knows that it has to load the Angular 2 components from “node\_modules” folder.  
Typescript compiler uses the configuration from “tsconfig.json” file. In the configuration we have one property termed as “moduleResolution. It has two values:-

* **Classic :-** In this mode typescript relies on “./” and “../” to locate folders.
* **Node :-** In this mode typescript first tries to locate components in “node\_modules” folder and if not found then follows the “../” convention to traverse to the folders.

In our tsconfig.json we have defined the mode as “node” this makes typescript hunt modules automatically in “node\_modules” folder. That makes the “import” of angular 2 components work.

Hide   Copy Code

{

{

....

....

"moduleResolution": "node",

....

....

}

}

So now that both the import statements are applied let us create the “CustomerComponent” and from that let’s expose “Customer” object to the UI with the object name “CurrentCustomer”.

Hide   Copy Code

export class CustomerComponent {

CurrentCustomer:Customer = new Customer();

}

As we said previously that component connects / binds the model to the HTML UI. So there should be some code which tells that “CustomerComponent” is bounded with HTML UI. That’s done by something termed as “Component MetaData Attribute”. A component metadata attribute starts with “@Component” which has a “templateUrl” property which specifies the HTML UI with which the component class is tied up with.

Hide   Copy Code

@Component({

selector: "customer-ui",

templateUrl: "../UI/Customer.html"

}

This attribute is then decorated on the top of the component. Below goes the full code.

Hide   Copy Code

@Component({

selector: "customer-ui",

templateUrl: "../UI/Customer.html"

})

export class CustomerComponent {

CurrentCustomer:Customer = new Customer();

}

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/27.jpg | Putting in simple words binding code is nothing but a simple typescript class which decorated by the “@Component” attribute which dictates that this typescript class is binded with which UI. |

Below goes the full code of the Angular component.

Hide   Copy Code

// Import statements

import {Component} from "@angular/core"

import {Customer} from '../Model/Customer'

// Attribute metadata

@Component({

selector: "customer-ui",

templateUrl: "../UI/Customer.html"

})

// Customer component class exposing the customer model

export class CustomerComponent {

CurrentCustomer:Customer = new Customer();

}

## Step 7:- Creating the Customer HTML UI – Directives and Interpolation

Now from the “CustomerComponent” , “Customer” is exposed via the “CurrentCustomer” object to UI. So in the HTML UI we need to refer this object while binding.

In the HTML UI the object is binded by using “Directives”. Directives are tags which direct how to bind with the UI.

For instance if we want to bind “CustomerName” with HTML textbox code goes something as shown below:-

* “[(ngModel)]” is a directive which will help us send data from the object to UI and vice versa.
* Look at the way binding is applied to the object. It’s referring the property as “CurrentCustomer.CustomerName” and not just “CustomerName”. Why ??. Because if you remember the object exposed from the “CustomerComponent” is “CurrentCustomer” object. So you need to qualify “CurrentCustomer.CustomerCode”.

Hide   Copy Code

<input type="text" />

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/28.jpg | * Round brackets indicate data sent from UI to object. * Square brackets indicate data is sent from object to UI. * If both are present then it’s a two way binding. |

There would be times when we would like to display object data on the browser. By using “{{“ braces we can display object data with HTML tags. In the below HTML we are displaying “CustomerName” mixed with HTML BR tag. These braces are termed as “INTERPOLATION”. If you see the dictionary meaning of interpolation it means inserting something of different nature in to something else.

In the below code we are inserting object data within HTML.

Hide   Copy Code

{{CurrentCustomer.CustomerName}}

Below goes the full HTML UI code with binding directives and interpolation.

Hide   Copy Code

<div>

Name:

<input type="text" [(ngModel)]="CurrentCustomer.CustomerName"><br /><br />

Code:

<input type="text" [(ngModel)]="CurrentCustomer.CustomerCode"><br /><br />

Amount:

<input type="text" [(ngModel)]="CurrentCustomer.CustomerAmount"><br /><br />

</div>

{{CurrentCustomer.CustomerName}}<br /><br />

{{CurrentCustomer.CustomerCode}}<br /><br />

{{CurrentCustomer.CustomerAmount}}<br /><br />

## Step 8:- Creating the Module

Module is a container or you can say it’s a logical grouping of components and other services.

So the first import in this module is the “CustomerComponent” component.

Hide   Copy Code

import { CustomerComponent }   from '../Component/CustomerComponent';

We also need to import “BrowserModule” and “FormsModule” from core angular. “BrowserModule” has components by which we can write IF conditions and FOR loop. “FormsModule” provides directive functionality like “ngModel”, expressions and so on.

Hide   Copy Code

import { BrowserModule } from '@angular/platform-browser';

import {FormsModule} from "@angular/forms"

We also need to create a typescript class “MainModuleLibrary”. At this moment this class does not have any code but it can have code which will provide component level logic like caching , initialization code for those group of components and so on.

Hide   Copy Code

export class MainModuleLibrary { }

To create a module we need to use import “NgModule” from angular core. This helps us to define module directives.

Hide   Copy Code

import { NgModule }      from '@angular/core';

“NgModule” has three properties:-

* **Imports: -** If this module is utilizing other modules we define the modules in this section.
* **Declarations: -** In this section we define the components of the modules. For now we only have one component ‘CustomerComponent”.
* **Bootstrap: -** This section defines the first component which will run. For example we can have “HomeComponent”, “CustomerComponent” and so on. But the first component which will run is the “HomeComponent” so that we need to define in this section.

Hide   Copy Code

@NgModule({

    imports: [BrowserModule,

             FormsModule],

    declarations: [CustomerComponent],

    bootstrap: [CustomerComponent]

})

Below goes the full code of Angular module which we discussed in this section.

Hide   Copy Code

import { NgModule }      from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import {FormsModule} from "@angular/forms"

import { CustomerComponent }   from '../Component/CustomerComponent';

@NgModule({

    imports: [BrowserModule,

             FormsModule],

    declarations: [CustomerComponent],

    bootstrap: [CustomerComponent]

})

export class MainModuleLibrary { }

## Step 9:- Creating the “Startup.ts” file

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/29.jpg | So we have created the UI, we have created the models, we have created components and these components are grouped in to modules. In an angular application you can have many modules.  So you need to define the startup module. So let’s create a “Startup.ts” file which will define the startup module. |

Below goes the “Startup.ts” file in which we have defined which module will be bootstrapped.

Hide   Copy Code

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { MainModuleLibrary } from '../Module/MainModuleLibrary';

const platform = platformBrowserDynamic();

platform.bootstrapModule(MainModuleLibrary);

## Step 10:- Invoking “Startup.ts” file using main angular page

So let us create a startup HTML page which will invoke the “Startup.ts”. Now in this page we will need to import four JavaScript framework files Shim , Zone , Meta-data and System JS as shown in the below code.

Hide   Copy Code

<script src="../../node\_modules/core-js/client/shim.min.js"></script>

<script src="../../node\_modules/zone.js/dist/zone.js"></script>

<script src="../../node\_modules/reflect-metadata/Reflect.js"></script>

<script src="../../node\_modules/systemjs/dist/system.src.js"></script>

Below are the use of JS files :-

|  |  |
| --- | --- |
| Shim.min.js | This framework ensures that ES 6 javascript can run in old browsers. |
| Zone.js | This framework ensures us to treat group of Async activities as one zone. |
| Reflect.js | Helps us to apply meta-data on Javascript classes. We are currently using @NgModule and @NgComponent as attributes. |
| System.js | This module will helps to load JS files using module protocols like commonjs , AMD or UMD. |

In this HTML page we will are calling the “systemjs.config.js” file. This file will tell system JS which files to be loaded in the browser.

Hide   Copy Code

<script src="../systemjs.config.js"></script>

<script>

System.config({

"defaultJSExtensions": true

});

System.import('startup').catch(function (err) { console.error(err); });

</script>

In the “import” we need to specify “startup” which will invoke “startup.js” file.

Hide   Copy Code

System.import('startup').catch(function (err) { console.error(err); });

Our customer screen in with the name “Customer.html”. So to load in to this screen we need to define a place holder. So in this place holder our Customer HTML page will load.

Hide   Copy Code

<customer-ui></customer-ui>

If you remember when we created the component class we had said to load the HTML page in a selector. So that selector is nothing but a tag (placeholder) to load our Customer page.

Hide   Copy Code

@Component({

    selector: "customer-ui",

    templateUrl: "../UI/Customer.html"

})

Below goes the full HTML page with all scripts and the place holder tag.

Hide   Copy Code

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8" />

</head>

<!-- 1. Load libraries -->

<!-- Polyfill(s) for older browsers -->

<script src="../../node\_modules/core-js/client/shim.min.js"></script>

<script src="../../node\_modules/zone.js/dist/zone.js"></script>

<script src="../../node\_modules/reflect-metadata/Reflect.js"></script>

<script src="../../node\_modules/systemjs/dist/system.src.js"></script>

<!-- 2. Configure SystemJS -->

<script src="../systemjs.config.js"></script>

<script>

System.config({

"defaultJSExtensions": true

});

System.import('startup').catch(function (err) { console.error(err); });

</script>

<body>

<customer-ui></customer-ui>

</body>

</html>

## Step 11:- Installing http-server and running the application

In order to run the application we need a web server. So go to integrated terminal and type “npm install http-server”. “http-server” is a simple, zero-configuration command-line http server which we can use for testing, local development and learning. For more details visit https://www.npmjs.com/package/http-server

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/30.jpg | To run this server we need to type “http” in the VS code integrated terminal as shown in the figure.  In case your 80 port is blocked you can run this server on a specific port using “http-server –p 99”. This will run this server over 99 port. |

|  |  |
| --- | --- |
| https://www.codeproject.com/KB/aspnet/1174685/31.jpg | So once the web server is running you can now browse to the main angular HTML page.  Main angular page means the page in which we have put the scripts, put the place holder , systemjs and so on.  Please note Customer.html is not the main page. This page will be loaded in the placeholder of main angular page.  Once the sites are running type in one of the textboxes and see the automation of binding output in expression. |